CONFIDENTIAL

UNITED STATES PATENT APPLICATION

FOR

FLEXIBLE ARCHITECTURE FOR IMAGE PROCESSING

Inventors:
David Kuo
Eric Anderson

Prepared by:
WAGNER, MURABITO & HAO
Two North Market Street
Third Floor
San Jose, California 95113

FLEXIBLE ARCHITECTURE FOR IMAGE PROCESSING

FIELD OF THE INVENTION

The present invention relates to the field of digital cameras, and more particularly relates to a platform-independent image processing architecture.

BACKGROUND OF THE INVENTION

Digital image devices, such as but not limited to digital cameras, are an increasingly popular means for capturing images (in other words, for taking pictures) and for processing the resulting image data.

In digital cameras, images are represented by data and stored either in the camera's memory or an external memory device from which they can be accessed by a user. A significant advantage to digital cameras is that users then have the capability to manipulate the image data in a number of ways. Users are able to operate on and modify the images, transfer them to other devices, incorporate them into documents, display them in a variety of formats, and the like. Thus, in comparison to conventional cameras, digital cameras introduce a variety of capabilities and enhancements.

The digital camera incorporates a central processing unit, memory, and many other features of a computer system. Accordingly, the digital camera is capable of concurrently running multiple software routines and subsystems to control and coordinate the various processes of the camera. One subsystem of particular interest is the image processing subsystem

that is used for analyzing and manipulating captured image data in a variety of ways, including linearization, defect correction, white balance, interpolation, color correction, image sharpening, and color space conversion. In addition, the subsystem typically coordinates the functioning and communication of the various image processing stages and handles the data flow between the various stages.

A problem exists in that there are many prior art makes and models of digital cameras incorporating different hardware components and software configurations (that is, "platforms") for image processing. One make of digital camera may utilize a software mode for image processing, another may implement a hardware mode, and others may implement a mode utilizing various degrees of hardware and software combinations. Typically, this problem is addressed in the prior art by employing an operating system written specifically for the image processing mode used by the particular digital camera platform.

However, this prior solution is problematic because an operating system designed for one digital camera platform is generally too specialized to function on another platform. Even in those cases where the image processing mode for one camera make/model shares a common aspect with another make/model, it is often not possible to utilize a common operating system because of their specialized natures.

Consequently, in the prior art it is necessary for manufacturers of digital cameras to, for example, develop, purchase or license more than one version of a digital camera operating system, e.g., one version for each type

of digital camera platform that the manufacturer offers. As the manufacturer develops additional makes and models employing various image processing modes, it is likely that different operating systems will also be needed. The need to integrate the development of the operating system with the development of the image processing mode introduces difficulties that could extend the lead time for introduction of a new product. For a manufacturer wishing to offer a number of different models of digital cameras, this can result in substantial effort and expense.

Similarly, the expense to a vendor of operating systems is also increased due to the need to support multiple digital camera platforms. In the prior art, different versions of operating systems increase development and manufacturing costs. The number of programmers needed to develop and support different operating systems is clearly a function of the number of operating systems that need to be provided. Increased costs are also associated with the support of multiple versions; for example, upgrades must be prepared for each version, and separate users' manuals must also be written and published.

Accordingly, it is desirable to provide an operating system and/or method that can be utilized with the image processing mode used by any digital camera platform. It is also desirable that the operating system and/or method be flexible enough to be adapted to foreseeable digital camera platforms. The present invention provides a novel solution to the above needs.

SUMMARY OF THE INVENTION

The present invention provides an operating system that can be utilized with the image processing mode on any digital camera platform. The present invention is also flexible enough to be adapted to foreseeable digital camera platforms.

The present invention is a system and a method for processing image data in a digital image device such as a digital camera. The present invention includes a bus, a central processing unit coupled to the bus, an image processing subsystem coupled to the central processing unit for processing the image data using a particular processing mode, a memory unit coupled to the bus, and a data storage element for storing the image data after image processing. The memory unit has stored therein an operating system for managing the image processing subsystem, and the memory unit also has a data structure for managing the image data for the image processing subsystem during image processing. The data structure provides an interface between the operating system and the image processing subsystem, such that the operating system is independent of the processing mode used by the image processing subsystem.

In the present embodiment, the present invention also includes a spooler element for transferring the image data into the data structure, a data line reader element for reading the image data from the spooler element, and a data line writer element for writing the image data to the data storage element. The data line writer element provides an interface between the image processing subsystem and the data storage element that

is independent of the processing mode used by the image processing subsystem.

In one embodiment, the processing mode used by the image processing subsystem includes a plurality of image processing modules and a JPEG (Joint Photographic Experts Group) software element. In another embodiment, the processing mode used by the image processing subsystem includes a digital signal processor and a JPEG hardware element. In yet another embodiment, the processing mode used by the image processing subsystem includes an image processing hardware system.

In the present embodiment, the present invention implements a method for processing image data in a digital image device that includes the following steps: a) creating a data structure corresponding to the image processing mode used by the image processing subsystem; b) initializing the spooler element; c) initializing the data line reader element; d) initializing the data line writer element; e) initializing the image processing subsystem and the data structure; f) forwarding the image data to the data structure using the spooler element; g) processing the image data using the processing mode used by the image processing subsystem; and h) writing the image data to the data storage element.

The present invention thus provides a flexible architecture for image processing within a digital image device (e.g., digital camera). The present invention permits the use of different modes of image processing while maintaining the same operating system, application program and

application program interface. That is, the image processing mode used in a particular digital camera platform is invisible to the operating system and to the application program. The present invention accomplishes this by defining data structures that are flexible enough that different imaging processing subsystems can be implemented without affecting the software/hardware architecture that surrounds the subsystem. The data structures provide a well-defined interface for entering and exiting the image processing subsystem without changing or perturbing the upstream and downstream elements of the digital camera.

These and other objects and advantages of the present invention will become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

FIGURE 1 is a block diagram exemplifying a digital camera in accordance with one embodiment of the present invention.

FIGURE 2 is a block diagram exemplifying the computer system used in a digital camera in accordance with one embodiment of the present invention.

FIGURES 3A and 3B are, respectively, a rear elevation view and a top elevation view of one embodiment of the digital camera of Figure 2 in accordance with the present invention.

FIGURE 4 is a diagram of one embodiment of the non-volatile memory of the digital camera of Figure 2 in accordance with the present invention.

FIGURE 5 is a diagram of the non-volatile memory of Figure 4 showing the image processing backplane and image processing modules in accordance with one embodiment of the present invention.

FIGURE 6 is a block diagram of the architecture utilized with a software image processing subsystem in accordance with the present invention.

FIGURE 7 is a block diagram illustrating the data path through the architecture of Figure 6 in accordance with the present invention.

FIGURE 8 is an illustration of the data structure associated with the architecture of Figure 6 in accordance with the present invention.

FIGURE 9 is a block diagram of the architecture utilized with a software/hardware image processing subsystem in accordance with the present invention.

FIGURE 10 is a block diagram illustrating the data path through the architecture of Figure 9 in accordance with the present invention.

FIGURE 11 is an illustration of the data structure associated with the architecture of Figure 9 in accordance with the present invention.

FIGURE 12 is a block diagram of the architecture utilized with a hardware image processing subsystem in accordance with the present invention.

FIGURE 13 is a block diagram illustrating the data path through the architecture of Figure 12 in accordance with the present invention.

FIGURE 14 is an illustration of the data structure associated with the architecture of Figure 12 in accordance with the present invention.

FIGURE 15 is a flowchart of a process for implementing the architectures of Figures 6, 9 and 12 in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Although the present invention will be described in the context of a digital camera, various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. That is, any digital image device which processes, displays and/or prints digital images, icons and/or other items, could incorporate the features described hereinbelow and that device would be within the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

In the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be recognized by one skilled in the art that the present invention may be practiced without these specific details or with equivalents thereof. In other instances, well known methods, procedures, components and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

The present invention is an image processing architecture that is designed to input a large amount of image data from a data store, operate upon that data, and output the processed data to another data store. The

actual processing may be performed by either software components, hardware components, or combinations of the two. The present invention is implemented in a digital camera operating system such as the Digita™ Operating Environment (OE) by FlashPoint Technology of San Jose, California.

In the Digita OE, the data store may either be in memory or on disk. The exact nature of the data store can be hidden from the processing module by using a well designed programming interface. From the perspective of the Digita OE, data from either form can be delivered to the processing module in a well described manner, which does not require any software changes.

For software components, the desire is to retrieve data into a buffer that can be referenced by the processing module. To accomplish this, the programming interface provides a routine that can be called to retrieve data from the data store. In the case of when the data store is in memory, this routine would copy the requested bytes of data from the data store memory into a memory buffer that can be referenced by the processing module. In the case of when the data store is on disk, this would read the data off disk into a memory buffer that can be referenced by the processing module.

For hardware components, the desire is to retrieve data through a direct memory access (DMA) operation, where data are transferred across a bus by hardware. To support this mode, the programming interface provides a routine that can be called to retrieve a pointer to a buffer that contains the data. In the case of when the data store is in memory, this

routine would simply return the address of the data store memory. In the case of when the data store is on disk, this routine would allocate a buffer, and read the data from disk into the newly allocated buffer, then return the address of the newly allocated buffer.

Combinations of hardware and software components are compatible with this design. The key point is for the processing module to identify whether it will be accessing data through the functional interface or through a pointer to memory. The Digita OE can respond to both types of requests without requiring any software changes.

Refer to Figure 1, which is a block diagram of one embodiment of digital camera 100 for use in accordance with the present invention. Digital camera 100 preferably comprises imaging device 114, system bus 116 and computer 110. Imaging device 114 includes an image sensor, such as a charged coupled device (CCD) or a complementary metal-oxide semiconductor (CMOS) sensor, for generating a set of raw image data representing a captured image of an object or person (e.g., object 112).

With reference now to Figure 2, system bus 116 provides connection paths between imaging device 114, optional power manager 342, central processing unit (CPU) 344, dynamic random-access memory (DRAM) 346, input/output interface (I/O) 348, non-volatile memory 350, and buffers/connector 352 that connect optional removable memory 354 to system bus 116.

CPU 344 may include a conventional processor device for controlling the operation of digital camera 100. In the preferred embodiment, CPU 344 is capable of concurrently running multiple software routines to control the various processes of digital camera 100 within a multi-threaded environment. For example, images may be captured at the same time that previously captured images are processed in the background to effectively increase the capture rate of the camera. In a preferred embodiment, CPU 344 runs an operating system capable of providing a menu-driven graphical user interface (GUI) and software image processing. An example of such software is the Digita™ Operating Environment by FlashPoint Technology of San Jose, California.

Continuing with reference to Figure 2, I/O 348 is an interface device allowing communications to and from computer 110. For example, I/O 348 permits an external host computer (not shown) to connect to and communicate with computer 110. I/O 348 also interfaces with a plurality of buttons and/or dials 404 and optional status liquid crystal display (LCD) 406 which, in addition to LCD screen 402, are the hardware elements of the digital camera's user interface 408.

Non-volatile memory 350, which may typically comprise a conventional read-only memory or flash memory, stores a set of computer-readable program instructions to control the operation of digital camera 100. Removable memory 354 serves as an additional image data storage area and is preferably a non-volatile device, such as a Flash disk, that is readily removable and replaceable by the user of digital camera 100 via buffers/connector 352.

With reference still to Figure 2, power supply 356 supplies operating power to the various components of digital camera 100. Power manager 342 communicates via line 366 with power supply 356 and coordinates power management operations for digital camera 100. In the preferred embodiment, power supply 356 provides operating power to main power bus 362 and also to secondary power bus 364. Main power bus 362 provides power to imaging device 114, I/O 348, non-volatile memory 350 and removable memory 354. Secondary power bus 364 provides power to power manager 342, CPU 344 and DRAM 346.

Power supply 356 is connected to main batteries 358 and also to backup batteries 360. In the preferred embodiment, a user of digital camera 100 user may also connect power supply 356 to an external power source. During normal operation of power supply 356, main batteries 358 provide operating power to power supply 356 which then provides the operating power to digital camera 100 via both main power bus 362 and secondary power bus 364. During a power failure mode in which main batteries 358 have failed (when their output voltage has fallen below a minimum operational voltage level), backup batteries 360 provide operating power to power supply 356 which then provides the operating power only to secondary power bus 364 of digital camera 100.

DRAM 346 is a contiguous block of dynamic memory that may be selectively allocated for various storage functions. DRAM 346 stores both raw and compressed image data and is also used by CPU 344 while executing the software routines used within computer 110. The raw image

data received from imaging device 114 are temporarily stored in several input buffers (not shown) within DRAM 346. Once the raw image data are processed, the data are stored in a frame buffer (not shown) for display on LCD screen 402. Finally, LCD controller 390 accesses DRAM 346 and transfers processed image data to LCD screen 402 for display.

Figures 3A and 3B are diagrams depicting the preferred hardware components of user interface 408 (Figure 2) of digital camera 100. Figure 3A is a back view of digital camera 100 showing LCD screen 402, four-way navigation control button 409, overlay button 412, menu button 414, and a set of programmable soft keys 416.

Figure 3B is a top view of digital camera 100 showing shutter button 418 and mode dial 420. The digital camera may optionally include status LCD 406, status LCD scroll button 422 and select button 424, sound record button 426, and zoom-in, zoom-out buttons 426a and 426b.

With reference back to Figure 3A, digital camera 100 is provided with several different operating modes for supporting various camera functions. The modes relevant to this description are capture or record mode for capturing images, and play mode for playing back the captured images on LCD screen 402. In capture mode, digital camera 100 supports the actions of preparing to capture an image and of capturing an image. In review mode, digital camera 100 supports the actions of reviewing camera contents, editing and sorting images, and printing and transferring images. In play mode, digital camera 100 allows the user to view screen-sized images in the orientation that the image was captured. Play mode

also allows the user to hear recorded sound associated to a displayed image, and to play back sequential groupings of images, which may comprise time lapse, slide show, and burst image images. The user preferably switches between the capture, review, and play modes.

With reference still to Figure 3A, to take a picture, digital camera 100 must be placed into capture mode. If LCD screen 402 is activated, then the camera displays to the user a "live view" of the object viewed through the camera lens on LCD screen 402 as a successive series of real-time frames. If LCD screen 402 is not activated, then the user may capture an image using a conventional optical viewfinder (not shown).

Continuing with reference to Figure 3A, during the execution of live view generation, frames of raw image data are sequentially captured by imaging device 114 (Figure 2) at a reduced resolution suitable for LCD screen 402, and the frames of raw image data are stored in DRAM 346 (Figure 2). The live view generation process then performs gamma correction and color conversion on the raw CCD data to convert the data into either a RGB or YCC color format which is compatible with LCD screen 402. (RGB is an abbreviation for Red, Green, Blue, and YCC is an abbreviation for Luminance, Chrominance-red and Chrominance-blue.) The raw image data are also processed for extracting exposure, focus, and white balance settings, and the like. After converting each frame of data to YCC (typically, YCC 2:2:2 format), the YCC image data are transferred to LCD screen 402 for display.

The live view frames generated during live view generation are displayed until the user decides to capture the image (i.e., take a picture). When the user presses the shutter button to capture an image, the imaged data are captured at a resolution set by the user, transformed into YCC 4:2:2 color space, compressed (e.g., JPEG), and stored as an image file. Live view then resumes to allow the capture of another image. The user may then either continue to capture images or switch digital camera 100 to play mode in order to play back and view the previously captured images on LCD screen 402. In play mode, the user may also hear any recorded sound associated with a displayed image.

Referring now to Figure 4, a diagram of one embodiment of non-volatile memory 350 of Figure 2 is shown. Non-volatile memory 350 includes application program 371, application program interface 372, toolbox 373, drivers 374, hardware abstraction layer 375, kernel 376, and system configuration 377.

Application program 371 includes program instructions for controlling and coordinating the various functions of digital camera 100 (Figure 2). Different application programs may be preferably loaded by the user into digital camera 100 depending on the planned uses of the camera. Application program interface 372 defines a standard interface between application program 371 and toolbox 373.

Toolbox 373 contains the various software routines used to control the various processes of digital camera 100. In general, toolbox 373 provides software routines for printing images, displaying images, organizing data,

and the like. Toolbox 373 includes a menu and dialogue manager, which comprises software routines that are used to provide information for controlling access to camera control menus and camera control menu items. The menus, in turn, enable communication between digital camera 100 and the user, and provide the user with access to the features of digital camera 100. Also included in toolbox 373 is a file formatter, consisting of software routines for creating an image file from processed image data, and a spooling manager.

Drivers 374 are for controlling various hardware devices within digital camera 100, such as the motors used to adjust the lens to change focus. Hardware abstraction layer (HAL) 375 defines a standard interface between toolbox 373/drivers 374 and the implementation hardware specific to the make and model of the digital camera. Kernel 376 provides basic underlying services for the operating system of digital camera 100. System configuration 377 performs initial start-up routines for digital camera 100, including the boot routine and initial system diagnostics.

Present Invention Architecture for Image Processing

The present invention is a flexible architecture for image processing in a digital image device (e.g., digital camera 100). As will be seen by the discussion below, the present invention permits the use of different modes of image processing while maintaining the same operating system, application program and application program interface (e.g., application program 371 and application program interface 372 of Figure 4). That is, the image processing mode used in a particular digital camera platform is invisible to the operating system and to application program 371.

The present invention accomplishes this by defining data structures that are flexible enough that different imaging processing subsystems can be implemented without affecting the software/hardware architecture that surrounds the subsystem. As will be seen, the present invention utilizes a well-defined interface that serves as a common point for entering and exiting the image processing subsystem without changing or perturbing the upstream and downstream elements of the digital camera.

The present invention provides the framework in which image processing can be accomplished in three different modes, depending on the degree of hardware and software assistance associated with the particular digital camera platform. The first mode, referred to herein as the "software architecture," implements an architecture that manages buffering within the architecture and that uses plug-in image processing modules. The second mode, referred to herein as the "software/hardware architecture," supports the use of function calls to read in and write out data and utilizes hardware elements for image processing. The third mode, referred to herein as the "hardware architecture," provides an address in memory for the input and output buffers in a direct memory access (DMA) environment, and uses an image processing hardware system. Each of these three modes are described in further detail below in conjunction with Figures 6 through 14.

The present invention therefore provides a framework that enables image processing to be accomplished for different digital camera platforms.

Hence, it is appreciated that the present invention may be implemented with image processing embodiments other than those described below.

Software Architecture

With reference now to Figure 5, in accordance with one embodiment of the present invention, non-volatile memory 350 also contains image processing backplane 510, which coordinates the functioning and communication of various image processing modules 520 and handles the data flow between these modules. Image processing modules 520 include selectable plug-in software routines that analyze and manipulate captured image data in a variety of ways, depending on the particular module(s) selected. In the present embodiment, three image processing modules are utilized. The first image processing module performs linearization, defect correction, and white balance. The second image processing module performs interpolation and color correction. The third image processing module performs sharpening and color space conversion. However, it is understood that different numbers and functions of image processing modules other than those just described may be used in accordance with the present invention.

Referring now to Figure 6, a block diagram of software architecture 600 is provided in accordance with the present invention. As explained above in conjunction with Figures 1 and 3A, when an image is captured by digital camera 100, the raw image data (typically in CCD format, e.g., CCD data 605) are stored in input buffers within DRAM 346.

In the present embodiment, image processing backplane 630 includes three plug-in image processing software modules IPM(1) 622, IPM(2) 624 and IPM(3) 626; however, it understood that a different number of image processing modules may be utilized in accordance with the present invention. Image processing backplane 630 also includes JPEG software 628. It should be appreciated that encoding systems other than JPEG may be used with the present invention.

The three image processing modules and JPEG software 628 function by reading lines of incoming data, processing the data, and then sending out lines of data. IPM(1) 622, IPM(2) 624, IPM(3) 626 and JPEG software 628 are loaded by image processing backplane 630 during startup of the digital camera. Image processing modules and JPEG software are well known in the art.

Line reader 620 and line writer 650 manage the flow of data into and out of IPM(1) 622, IPM(2) 624, IPM(3) 626 and JPEG software 628. In this embodiment, line reader 620 is a software element that provides a functional interface for retrieving lines of data from CCD data 605. Image processing backplane 630 uses this interface to retrieve lines of data that are fed into IPM(1) 622, which in turn feeds into IPM(2) 624, which in turn feeds into IPM(3) 626, which in turn feeds into JPEG software 628. In the case where the digital camera has an orientation sensor and the image is captured with the camera rotated from a purely horizontal or vertical orientation, then line reader 620 feeds the lines of data in a rotated fashion so that the image on the disk (e.g., removable memory 354 of Figure 3A) is also rotated.

In the present embodiment, water mark 980 includes software used to impress on the image data such information as the time and date. However, it is appreciated that in other embodiments of the present invention, the water marking function may not be used.

Continuing with reference to Figure 6, line writer 650 is a software element that writes lines of data to disk 660 (e.g., RAM disk or Flash disk).

With reference now to Figure 7, the data path through software architecture 600 is illustrated. Lines of data are fed through spooler 700; as used herein, a spooler is a process or device for transferring data from one process or device to a second process or device. Spooler 700 is comprised of RAM disk 1 spooler 701, Flash disk 1 spooler 702, image processing spooler 703, RAM disk 2 spooler 704 and Flash disk 2 spooler 705; however, it is understood that a different spooler configuration may be used in accordance with the present invention. In accordance with the present invention, spooler 700 initializes line reader 620 and line writer 650.

The lines of data fed through spooler 700 can be from any source, e.g., the data can come from memory (e.g., an image buffer), from the RAM disk (as a file), or from the Flash disk (as a file). In accordance with the present invention, image processing spooler 703 is used to provide the data to line reader 620. As indicated by the illustration, line reader 620, image processing backplane 630, JPEG software 628 and line writer 650 are situated below application program interface 372 on the data path. Hence, in accordance with the present invention, the image processing subsystem

(e.g., image processing backplane 630 including JPEG software 628) is invisible to the operating system and to application program 371 (Figure 5).

Refer now to Figure 8, which illustrates data structure 800 associated with software architecture 600. Input buffer 810 is a buffer that is created and used only when data are not spooled, in which case the data are loaded from DRAM 346. Otherwise, lines of image data are read by line reader 620 from, for example, a RAM disk or a Flash disk via spooler 700 as shown in Figure 7.

The lines of data are read into input/output buffer 820, which serves as the output buffer for line reader 620 and the input buffer for IPM(1) 622. In the present embodiment, input/output buffer 820 can hold only one line of data. Thus, when the next line of data is received by input/output buffer 820, the line of data already in that buffer is driven into IPM(1) 622.

After processing by IPM(1) 622 (e.g., linearization, defect correction, and white balance), the line of data is transferred to input/output buffer 830, which serves as the output buffer for IPM(1) 622 and the input buffer for IPM(2) 624. In the present embodiment, input/output buffer 830 is capable of holding up to five lines of data. The lines of data are not forwarded from input/output buffer 830 to IPM(2) 624 until the buffer is full. When the buffer is full, the next line of data received by input/output buffer 830 drives one line of data into IPM(2) 624.

After processing by IPM(2) 624 (e.g., interpolation and color correction), the line of data is transferred to input/output buffer 840, which

serves as the output buffer for IPM(2) 624 and the input buffer for IPM(3) 626. In the present embodiment, input/output buffer 840 is capable of holding only one line of data. Thus, in the same manner as above, when the next line of data is received by input/output buffer 840, the line of data already in that buffer is driven into IPM(3) 626.

After processing by IPM(3) 626 (e.g., sharpening and color space conversion), the line of data is transferred to input/output buffer 850, which serves as the output buffer for IPM(3) 626 and the input buffer for JPEG software 628. JPEG software 628 accepts lines of data as input. As data are buffered and processed through JPEG software 628, a stream of bytes that represent compressed data is forwarded to file buffer 860 then to line writer 650, which writes the data to, for example, the RAM disk or the Flash disk.

Software/Hardware Architecture

With reference now to Figure 9, a block diagram of software/hardware architecture 900 is provided for a different image processing embodiment in accordance with the present invention. As explained above in conjunction with Figures 1 and 3A, when an image is captured by digital camera 100, the raw image data (e.g., CCD data 605) are stored in input buffers within DRAM 346.

In this embodiment, image processing backplane 630 includes a hybrid of software and hardware components. Relative to software architecture 600 (Figure 6) described above, in software/hardware architecture 900 the software image processing modules are replaced by digital signal processor (DSP) 922 and the JPEG software is replaced by

JPEG hardware 924. Digital signal processors and JPEG hardware are well known in the art. It should be appreciated that encoding systems other than JPEG may be used with the present invention.

In the present embodiment, water mark 980 includes software used to impress on the image data such information as the time and date. However, it is appreciated that in other embodiments of the present invention, the water marking function may not be used.

The software elements line reader 620 and line writer 650 are retained, and they function as described above in conjunction with Figures 6, 7 and 8. These elements serve as part of the common interface between the operating system and the image processing subsystem that is independent of the mode of image processing that is used.

With reference now to Figure 10, the data path through software/hardware architecture 900 is illustrated. In the same manner as described for software architecture 600 (see discussion in conjunction with Figure 7), lines of data are fed through spooler 700. The lines of data are from any source (e.g., memory, RAM disk, or Flash disk), and are fed using image processing spooler 703 to line reader 620. The data can include thumbnail, screennail, or image information. In accordance with the present invention, spooler 700 initializes line reader 620 and line writer 650.

Continuing with reference to Figure 10, hardware accelerator 1040 comprises DSP 922 and JPEG hardware 924 of Figure 9. Line reader 620, hardware accelerator 1040, and line writer 650 are situated below

application program interface 372 on the data path. Hence, in accordance with the present invention, the image processing subsystem (e.g., hardware accelerator 1040) is invisible to the operating system and to application program 371 (Figure 5).

Refer now to Figure 11, which illustrates data structure 1100 associated with software/hardware architecture 900. Input buffer 1110 is a buffer that is created and used only when data are not spooled, in which case the data are loaded from DRAM 346. Otherwise, in the same manner as described above in conjunction with software architecture 600 of Figure 6, lines of data are read by line reader 620 from, for example, the RAM disk or the Flash disk via spooler 700 as shown in Figure 10.

The lines of data are read into input/output buffer 1120, which serves as the output buffer for line reader 620 and the input buffer for DSP 922. After processing by DSP 922, the data are forwarded to ping-pong buffers A and B 1130 and from there to JPEG hardware 924. DSP 922 and JPEG hardware 924 can be run in parallel (at the same time). Thus, one of the ping-pong buffers (e.g., buffer A) is filled and the data therein are then fed to JPEG hardware 924. JPEG hardware 924 operates on these data while the other ping-pong buffer (e.g., buffer B) is filled. When JPEG hardware 924 is finished with the data from buffer A, it begins to operate on the data in buffer B, and in the meantime buffer A is refilled. Water mark 980 is impressed on the image data in ping-pong buffers A and B 1130.

After processing by JPEG hardware 924, the data are forwarded to input/output buffer 1140, then to line writer 650, which writes the data to,

for example, the RAM disk or the Flash disk. Line writer 650 provides an interface between the image processing subsystem and the downstream elements of the digital camera that is independent of the image processing mode being used.

Hardware Architecture

With reference now to Figure 12, a block diagram of hardware architecture 1200 is provided for a different image processing embodiment in accordance with the present invention. As explained above in conjunction with Figures 1 and 3A, when an image is captured by digital camera 100, the raw image data (e.g., CCD data 605) are stored in input buffers within DRAM 346.

In this embodiment, the image processing backplane uses only hardware (e.g., image processing backplane hardware 1230) for image processing. Relative to software architecture 600 (Figure 6) described above, in hardware architecture 1200 the software image processing modules and the JPEG software are replaced by an image processing hardware system. Relative to software/hardware architecture 900 (Figure 9) described above, the software-assisted digital signal processor and JPEG hardware are replaced by an image processing hardware system. In addition, as will be described in detail below, the functions performed by line reader of hardware architecture 1200 are extended to include additional functions relative to line reader 620 of software architecture 600 and software/hardware architecture 900.

In this embodiment, CCD data 605 is read from memory using direct memory access (DMA) and processed by image processing backplane hardware 1230. Image processing backplane hardware 1230 is exemplified by the image processing hardware system described in the patent application assigned to the assignee of the present invention and entitled "Image Processing System for High Performance Digital Imaging Devices;" by Eric C. Anderson; Serial Number 09/081,694; Attorney Docket Number FSPT-P156; filed on May 19, 1998; and herein incorporated by reference. Other image processing hardware systems known in the art may also be used in accordance with the present invention.

With reference now to Figure 13, the data path through hardware architecture 1200 is illustrated. As stated above, hardware architecture 1200 is DMA-based. Lines of data are passed through spooler 700; however, in this case, the stages of spooler 700 (specifically, RAM disk spooler 701 and Flash disk spooler 702) are each turned off so that the data flow through the spooler without being spooled. Thus, the data are fed through spooler 700 very quickly and the associated latency is not significant. Lines of data are still read off of the third stage of spooler 700 (e.g., image processing spooler 703). By retaining the spooler in hardware architecture 1200 in this manner, the same application program interface (e.g., application program interface 372) is maintained and the image processing subsystem remains invisible to the operating system and to application program 371 (Figure 5).

Refer now to Figure 14, which illustrates data structure 1300 associated with hardware architecture 1200. In this embodiment, data has

to be in memory (e.g., DRAM 346); thus, if the data are previously spooled, they must be despooled and brought back to memory. In accordance with the present invention, spooler 700 initializes line reader 1320 and line writer 650.

In this embodiment, the functions of line reader 1320 are extended relative to the functions of line reader 620 of software architecture 600 and software/hardware architecture 900 (Figures 6 and 9, respectively). Line reader 1320 functions to manage the data so that the data are properly located for processing. That is, line reader 1320 searches memory for the data and identifies the location of the image data to be processed. If the data are not in memory, then line reader 1320 directs memory to be allocated (e.g., input buffer 1310), reads the data off of the disk (e.g., the RAM disk), and places the data in input buffer 1310. Line reader 1320 thereby provides the memory location so that DMA channels can be set up. If the data are previously spooled, line reader 1320 directs that the data be despooled.

Continuing with reference to Figure 14, once input buffer 1310 is created by line reader 1320, the digital camera's operating system then sets up DMA engine 1430 and output buffer 1350, and executes image processing. Line writer 650 writes the data to the RAM disk, the Flash disk, or memory (e.g., DRAM 346). Line writer 650 provides an interface between the image processing subsystem and the downstream elements of the digital camera that is independent of the image processing mode being used.

If the water mark function is being used, the hardware operates in two phases with the water mark being impressed between the two phases. DMA engine 1430 sends data to and retrieves data from image processing backplane hardware 1230 in the first phase and places the partially processed data into water mark buffer 1340. The water mark is placed on the data in water mark buffer 1340. DMA engine 1430 then sends the data back to image processing backplane hardware 1230 for the second phase of processing. It is appreciated that other embodiments of the present invention may not use the water marking function.

Method of Present Invention Architecture for Image Processing

Figure 15 is a flow chart illustrating process 1500 for implementing the present invention architecture for image processing (e.g., software architecture 600, software/hardware architecture 900, and hardware architecture 1200). Process 1500 is implemented by the digital camera's operating system (e.g., Digita™) executed by CPU 344 of Figure 2. It should be appreciated that process 1500 is an exemplary process and that the present invention is not limited by the embodiment discussed below.

In step 1510, application program 371 initializes the spooler (e.g., spooler 700 of Figures 7, 10 and 13).

In step 1520, spooler 700 initializes the line reader and the line writer (e.g., line reader 620, line reader 1320, and line writer 650 of Figures 8, 11 and 14). To initialize the line reader, spooler 700 specifies, for example, the rows and columns of the input buffer, the Bayer pattern, the orientation of the data, and the address of the input buffer. To initialize the line writer,

spooler 700 specifies the output buffer to be used or the file extension (for example, for the RAM disk, Flash disk, or some another location within the digital camera).

In step 1530, spooler 700 initializes the image processing subsystem and the data structure associated with the image processing mode. In accordance with the present invention, software architecture 600 and data structure 800, software/hardware architecture 900 and data structure 1100, or hardware architecture 1200 and data structure 1400 may be utilized. In the present embodiment, spooler 700 disables the RAM disk spooler and the Flash disk spooler (e.g., RAM disk 1 spooler 701 and Flash disk 1 spooler 702) if hardware image processing is utilized.

In step 1540, the data are passed from the input buffer to spooler 700. As described above, spooler 700 is used to spool the data for software architecture 600 and software/hardware architecture 900, but the stages of spooler 700 are turned off in hardware architecture 1200. However, by retaining spooler 700 in each of these architectures, process 1500 is sufficiently flexible to be used with each of these architectures.

In step 1550, spooler 700 and the image processing subsystem are executed to convert raw image data into a compressed image file. The image processing subsystem used is either the software implementation of software architecture 600, the software/hardware implementation of software/hardware architecture 900, or the hardware implementation of hardware architecture 1200. In this embodiment, spooler 700 spools the data according to which of its stages are turned on, forwards data to the

line reader, and directs the image processing subsystem to being processing. The process data are then written to memory, the RAM disk, or the Flash disk.

In summary, the present invention is a flexible architecture for image processing within a digital image device such as a digital camera. The present invention permits the use of different types of image processing subsystems within a single operating system. The type of image processing subsystem is invisible to the particular operating system and application program, and in effect one image processing subsystem could be swapped out and another swapped in without the operating system and application program having to recognize that a change occurred.

The present invention provides the framework in which the image processing can be accomplished using software, hardware, or a combination of software and hardware. The present invention accomplishes this by defining data structures that are flexible enough that different imaging processing subsystems can be implemented without affecting the software/hardware architecture that surrounds the subsystem. The present invention utilizes a well-defined interface that suitably provides a common point for entering and exiting the image processing subsystem without changing or perturbing the upstream and downstream elements of the digital camera (conversely, the interface provides a common point at which the image processing subsystem taps into the data path of the digital camera without perturbing the data path). Thus, the present invention provides an operating system that can be used

with an digital camera platform and is adaptable for use with foreseeable platforms.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.